이진 표현을 이용한 효율적인 연관 규칙 탐사 알고리즘

김원영*, 최원길*, 김응모*
*성균관대학교 정보통신공학부
e-mail:bluemint88@gmail.com

An Efficient Algorithm for Mining Association Rules using a Binary Representation

Won-Young Kim*, Won-Gil Choi*, Ung-Mo Kim*
*School of Information and Communication Engineering, Sungkyunkwan University

요 약

오늘날 지식을 기반으로 하는 고도의 정보사회로 나아가는 시점에서 우리는 대량의 데이터 속에서 필요한 지식을 찾아내는 것에 초점을 모으게 되었다. 따라서 대량의 데이터 속에서 필요한 지식을 자동으로 찾아내는 데이터 마이닝에 대한 연구가 활발히 진행되고 있다. 데이터 마이닝은 대용량의 데이터를 대상으로 하기 때문에 정확도뿐만이 아니라 소요시간도 중요하기 때문에 성능 향상을 위한 알고리즘들이 많이 개발되었다. 데이터 마이닝의 성능을 향상시키기 위해서 가장 좋은 방법이 데이터베이스의 스캔의 횟수를 줄이는 것이다. 본 논문에서는 연관 규칙 탐사에서 빈발 항목 집합을 찾아내는 부분을 이진 표현을 이용하여 좀 더 성능을 향상시킬 수 있는 알고리즘을 제안한다.

1. 서론

오늘날 우리는 마케팅 데이터, 금융 데이터, 과학 데이터, 의학 데이터, 인구통계 데이터 등 수많은 데이터의 범람 속에 살고 있다. 그리고 지식을 기반으로 하는 고도의 정보사회로 나아가는 시점에서 우리의 관심은 대량의 정보관리와 검색과 같은 단순작업에서 벗어나 대량의 데이터 속에서 필요한 지식을 자동으로 찾아내는 방법에 초점을 모으게 되었다. 따라서 우리는 데이터를 자동으로 분석, 분류, 요약하고 그 안에 있는 추세를 자동으로 찾아서 특성화하고, 비정상적인 데이터를 자동으로 알려주는 방법들을 찾아야 한다. 이러한 방법들은 그동안 인공지능, 통계, 데이터베이스 등 다양한 학문 분야에서 연구되어 오던 지식을 추출하던 기법들을 모아 오늘날 데이터 마이닝 이라고 부르는 학문 분야로 정립되었다. 데이터 마이닝은 대용량의 데이터를 대상으로 하기 때문에 정확도에 대한 연구와 함께 성능을 향상시켜 소요시간을 줄이는 연구도 활발히 이루어지고 있다. 이에 따라 성능을 향상시키는 알고리즘들이 많이 만들어졌다. 데이터 마이닝의 성능에는 디스크 I/O에 의한 CPU의 오버헤드가 많은 영향을 미치는데 디스크 I/O의 오버헤드를 줄이기 위해서는 데이터베이스 스캔 횟수를 줄여야한다. 데이터베이스의 스캔의 횟수를 줄임으로써 데이터 마이닝의 성능을 향상시키는데 가장 큰 효과를 얻을 수 있다[3][9].

데이터 마이닝의 기법중 하나인 연관 규칙 탐사는 최소 지지도를 만족하는 높은 지지도를 가진 빈발 항목 집합을 찾아내어 그 빈발 항목 집합으로부터 최소 신뢰도를 만족하는 연관 규칙을 분석해내는 두 단계로 이루어져있다[1][8]. 연관 규칙 탐사에서 빈발 항목 집합을 찾아내는 단계에서 어려움이 존재하고, 대부분의 연관 규칙 탐사 알고리즘은 빈발 항목 집합을 찾아내는 방법에 대해서 성능을 향상시키는 것에 초점을 맞추고 있다[8]. 본 논문에서는 이진 표현을 이용하여 좀 더 효율적으로 빈발 항목 집합을 찾아내는 알고리즘을 제안한다.

본 논문은 다음과 같이 구성된다. 2장에서 연관규칙 탐사에 대해 알아보고 기존의 대표적인 알고리즘들을 설명한다. 3장에서는 새롭게 제안하는 이진 표현을 이용한 연관 규칙 탐사 알고리즘에 대해서 설명하고, 4장에서 결론 및 향후 연구 과제를 제시한 후 본 논문을 마친다.

2. 관련 연구

연관 규칙 탐사의 전형적인 예는 장바구니 분석(market basket analysis)이다. 장바구니 분석은 트랜잭션이 빈번하게 발생하는 물품 판매를 통해 만들어지는 [표 1]의 데이터베이스 같은 트랜잭션 데이터베이스를 분석하는 것을 말한다.

TID	항목
T100	I_1, I_2, I_5
T200	I_2, I_4, I_6
T300	I_2, I_3
T400	I_1, I_2, I_6
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3, I_6
T800	I_1, I_2, I_5, I_6
T900	I_1, I_2, I_3

<표 1> 트랜잭션 데이터베이스

2.1. 빈발 항목 집합의 정의

판매되는 항목들의 집합 $I=\{I_1, I_2, I_3, ..., I_n\}$로 정의된다. 트랜잭션 T는 소비자가 한 번에 구입하는 물품의 항목들의 집합이고 트랜잭션 T는 항목들의 집합 I의 부분집합이다. 데이터베이스 D를 n개의 트랜잭션들의 집합이라 하고 각각의 트랜잭션은 TID라는 고유한 번호가 부여된다. 트랜잭션 T가 I의 부분집합인 X를 포함한다면 T가 X를 지지한다(support)고 하고 supp(X)로 나타낸다. X의 지지도가 사용자가 정한 최소 지지도(minimum support) min_supp 보다 크거나 같으면 X가 빈발 항목 집합이라고 한다.

다음의 세 가지 특성들은 [1][2][3]에서 얻어진 특성들로 대부분의 연관 규칙 탐사 알고리즘에서 사용한다[6]. 본 논문에서 제시하는 알고리즘 또한 다음의 특성들을 사용한다.

(1) 항목집합 X, Y에 대하여 $X \subseteq Y$ 이면, Y를 지지하는 D의 모든 트랜잭션들이 X도 지지하므로 supp(X) ≥ supp(Y)이다.

(2) 항목집합 X가 D의 최소 지지도를 만족하지 못하면 (1)에 의해서 X의 모든 상위집합들은 빈발하지 않을 것이다.

(3) 항목집합 X가 D의 최소 지지도를 만족하여 빈발하다면 X의 모든 부분집합들은 빈발할 것이다. 하지만 그 역은 성립하지 않는다.

2.2. 연관 규칙의 정의

연관 규칙은 어떤 사건이 일어나면 다른 사건이 일어나는 연관성을 말하며, X와 Y를 항목들의 집합이라고 했을 때 R: X → Y 의 형식을 가진다. 이 때 X와 Y는 서로 같은 원소를 갖지 않는 항목집합이다. 한 트랜잭션이 X를 지지한다면, 어떤 확률에 의해 Y를 지지한다는 것을 예측할 수 있는 것이 연관 규칙이다. 여기서 어떤 확률을 연관 규칙의 신뢰도(confidence)라 하고 conf(R)로 나타낸다. 신뢰도는 X를 지지하는 트랜잭션에 대하여 Y또한 지지할 조건부 확률로 정의된다. 연관 규칙의 신뢰도는 X에 대하여 Y가 얼마나 자주 적용할 수 있는지를 나타내고, 지지도는 그 규칙 전부가 얼마나 빈번하게 발생하는 것인지를 보여준다. 따라서 데이터베이스에서 충분한 지지도와 신뢰도를 가져야 그 연관 규칙이 사용자에게 필요한 정보가 되는 것이다[6]. 연관 규칙에 대한 추가적인 특성은 [4]에서 참조할 수 있다.

2.3. 기존의 알고리즘

Apriori 알고리즘[1]은 연관 규칙 탐색 알고리즘 중에서 가장 기본적이고 유용한 알고리즘이다. Apriori 알고리즘은 k번째 항목 집합이 k+1번째 항목 집합을 찾기 위해 반복해서 사용되는 연역적 접근 방법을 사용한다. k+1번째 항목 집합까지 더 이상의 빈발 항목 집합이 없을 때까지 진행이 되는데 다음 항목을 찾을 때마다 계속 데이터베이스를 스캔하면서 진행된다. Apriori 알고리즘은 빈발 항목 집합을 수준단위로 생성하는 것을 효과적으로 개선하기 위해서 위에서 설명한 (1)(2)(3)의 특성을 기반으로 하는 Apriori 성질을 사용하여 탐색공간을 감소시킨다.

Apriori 알고리즘의 효율성을 향상시키기 위해 변화된 여러 가지 알고리즘들이 제안되었는데 그 방법은 다음과 같다.

DHP(Direct Hashing & Prunning) 알고리즘[5]은 기존의 Apriori 알고리즘에서 후보 빈발 항목 집합들의 수를 줄여서 지지도를 계산할 때 항목 집합에 대한 탐색시간을 줄였다. 그리고 항목 집합의 길이가 길어지면서 전체 고려대상이 되는 트랜잭션 수를 줄여서 데이터베이스 스캔 횟수를 줄였다.

Apriori와 DHP 모두 각 단계마다 데이터베이스를 한 번씩 스캔해야 하는데 [7]에서 제안하는 알고리즘은 몇 단계분의 지지도를 한번의 데이터베이스 스캔으로 계산해낸다. [7]의 알고리즘은 데이터베이스의 스캔 횟수를 줄여서 성능을 향상시킨 알고리즘이다.

위의 알고리즘들은 성능에 큰 영향을 미치는 두 가지 중요한 비용이 소비한다. 후보 항목 집합 생성과 반복적인 데이터베이스 스캔이다. 이에 [8]에서 후보 항목 집합을 생성하지 않고 빈발 항목 집합을 발견하는 알고리즘을 제안한다. [8]은 데이터베이스를 빈발 패턴 트리(FP-트리)로 압축을 하여 빈발 패턴 증가(FP-증가)라고 하는 기법을 사용한다. 기존의 알고리즘들의 성능에 대한 비교는 [6][9]에서 참조할 수 있다.

본 논문에서는 이진 표현을 이용하여 후보 항목 집합을 만들지 않고, 데이터베이스의 스캔 횟수를 줄이면서 빈발 항목 집합을 찾는 알고리즘을 제안한다.

3. 이진 표현을 이용한 연관 규칙 탐사 알고리즘

표 1의 트랜잭션 데이터베이스를 이용해서 최소지지도 min_supp는 2로 정하고 제안하는 알고리즘을 이용해서 빈발 항목 집합을 구한다. 본 논문에서 제안하는 알고리즘은 다음과 같은 단계로 구성된다.

(1) 트랜잭션 데이터베이스를 이진 형식으로 변환한다.

(2) 각 항목에 대해서 count를 구해서 사용자가 정한 최소 지지도보다 작으면 그 항목을 지운다.

(3) 이진 형식으로 변환된 각각의 트랜잭션들에 대해서 I_num과 TID_val을 구한다.

(4) 트랜잭션의 I_num가 큰 순서대로 TID_val로 XOR 연산을 이용해서 트랜잭션의 지지도를 계산하여 빈발 항목 집합을 찾아낸다.

3.1. 이진 형식으로 변환

[표 1]의 장바구니 데이터는 [표 2]처럼 이진 형식으로 표현할 수 있다. [표 2]의 각 행은 한 트랜잭션에 대응되고, 각 열은 한 항목에 대응된다. 한 항목은 이진 변수로 취급하고, 만약 그 항목이 트랜잭션에 있으면 그 값은 1이 되고 그렇지 않으면 0이 된다.

TID	I_1	I_2	I_3	I_4	I_5	I_6
T100	1	1	0	0	1	0
T200	0	1	0	1	0	1
T300	0	1	1	0	0	0
T400	1	1	0	0	0	1
T500	1	0	1	0	0	0
T600	0	1	1	0	0	0
T700	1	0	1	0	0	1
T800	1	1	0	0	1	1
T900	1	1	1	0	0	0

<표 2> 이진 형식으로 변환한 트랜잭션 데이터베이스

3.2. 변수 계산

이진 형식으로 변환한 트랜잭션 데이터베이스에 대해서 본 논문에서 정의하는 count, I_num, TID_val 변수들을 계산한다.

TID	I_1	I_2	I_3	I_4	I_5	I_6
T100	1	1	0	0	1	0
T200	0	1	0	1	0	1
T300	0	1	1	0	0	0
T400	1	1	0	0	0	1
T500	1	0	1	0	0	0
T600	0	1	1	0	0	0
T700	1	0	1	0	0	1
T800	1	1	0	0	1	1
T900	1	1	1	0	0	0
count	6	7	5	1	2	4

<표 3> count를 계산한 트랜잭션 데이터베이스

count는 각각의 행에 대한 합이다. 즉, 데이터베이스 내에서 각 항목에 대한 개수가 된다. [표 3]처럼 각 항목에 대해서 count를 구한다. 그리고 각 항목의 count에 대해서 min_supp와 비교를 해서 min_supp보다 작은 항목은 삭제를 한다. min_supp보다 작은 항목은 드물게 발생하는 항목이기 때문에 사용자의 관심을 가지기 못하기 때문이다. I_num는 각각의 열에 대한 합이다. 즉, 각 트랜잭션의 항목에 대한 이진값을 모두 더한 값으로 그 트랜잭션이 가지고 있는 항목의 개수를 뜻한다. TID_val는 이진 형식으로 변환된 데이터베이스의 이진값으로 동일한 항목을 가진 트랜잭션이면 동일한 값을 가지게 되므로 그 트랜잭션의 지지도가 높아진다. 예를 들어 [표 4]에서 T400 트랜잭션의 TID_val는 11001가 된다.

[표 3]은 [표 2]에서 count를 구한 결과를 나타낸다. [표 4]는 count를 구한 후, min_supp보다 작은 count를 가진 항목을 삭제하고 I_num과 TID_val를 구한 결과이다.

TID	I_1	I_2	I_3	I_5	I_6	I_num	TID_val
T100	1	1	0	1	0	3	11010
T200	0	1	0	0	1	2	01001
T300	0	1	1	0	0	2	01100
T400	1	1	0	0	1	3	11001
T500	1	0	1	0	0	2	10100
T600	0	1	1	0	0	2	01100
T700	1	0	1	0	1	3	10101
T800	1	1	0	1	1	4	11011
T900	1	1	1	0	0	3	11100
count	6	7	5	2	4		

<표 4> I_num을 계산한 트랜잭션 데이터베이스

3.3. 빈발 항목 집합 발견

[표 4]의 I_num과 TID_val를 바탕으로 빈발 항목 집합을 찾아낼 수 있다. 두 변수를 이용하는 규칙이 다음과 같다.

(1) I_num이 가장 큰 트랜잭션의 TID_val부터 시작한다. 비교 대상이 된 I_num이 동일한 트랜잭션들에 대해서는 TID 순서대로 순서를 정한다. 대상 트랜잭션의 지지도는 1이 되고, 동일한 I_num을 가진 트랜잭션에 대해서 XOR(Exclusive OR) 연산을 해서 1의 개수가 0이 나오면 그 트랜잭션의 지지도를 1씩 증가시킨다.

(2) 동일한 I_num의 트랜잭션들에 대해서 XOR 연산이 끝나면 I_num이 큰 트랜잭션들에 대해서도 XOR 연산을 한다. 이때는 XOR 연산을 해서 1의 개수가 I_num의 차와 동일하면 그 트랜잭션의 지지도를 1씩 증가시킨다. 만약 I_num이 큰 트랜잭션이 없을 경우에는 지지도 계산이 끝난 트랜잭션들 중에 min_supp를 만족하는 트랜잭션이 있는지 비교한다.

(3) (2)단계에서 min_supp를 만족하는 트랜잭션이 있으면 그 트랜잭션의 항목 집합이 빈발 항목 집합이 된다. 빈발 항목 집합을 찾지 못했을 경우에는 I_num이 작은 I_num의 트랜잭션들로 (1)과 (2)의 단계를 반복해서 빈발 항목 집합을 찾는다.

XOR 연산은 배타적 논리합으로 연산자 기호는 ^로 표시한다. 두 비트를 비교해서 같으면 0, 다르면 1을 반환하는 연산자이다. 따라서 연산 결과에서 1의 개수가 I_num의 차와 같으면 I_num이 작은 트랜잭션이 I_num이 큰 트랜잭션에 포함이 된다는 것을 알 수 있다. 예를 들어 11010^11011 의 연산 결과는 00001 이 되므로 1의 개수가 두 트랜잭션의 I_num의 차와 같으므로 11010은 11011에 포함 된다고 할 수 있다.

[표 4]에서 T800의 I_num이 가장 크므로 T800의 지지도를 1로 하고, 동일 I_num의 트랜잭션을 찾아본다. 동일 I_num의 트랜잭션이 없기 때문에 T800의 지지도와 min_supp를 비교한다. T800의 지지도는 min_supp를 만족하지 못하므로 빈발 항목 집합이 아니다. 다음 크기의 I_num을 가진 트랜잭션들(T100, T400, T700, T900)에 대

해서 연산을 시작한다. TID 순서대로 T100부터 연산을 시작한다. T100의 지지도는 1이 되고 TID_val를 T400, T700, T900의 TID_val와 각각 XOR 연산을 해서 1의 개수가 0인지 비교한다. 동일 I_num을 가진 트랜잭션들의 TID_val와 XOR 연산이 끝나면 더 큰 I_num을 가진 T800의 TID_val와 XOR 연산을 한다. XOR 연산 결과의 1의 개수가 I_num의 차이인 1이 나오므로 T100의 지지도가 1 증가한다. T400, T700, T900의 트랜잭션들도 T100의 연산 과정을 반복하면 [표 5]와 같은 결과가 나온다. [표 5]에서 보면 T100과 T400의 지지도가 min_supp를 만족하므로 빈발 항목 집합이 된다. [표 4]에서의 T100과 T400의 항목 집합은 각각 $[I_1, I_2, I_5]$, $[I_1, I_2, I_6]$이다. 따라서 표 1의 트랜잭션 데이터베이스의 빈발 항목 집합은 $[I_1, I_2, I_5]$, $[I_1, I_2, I_6]$가 된다.

TID	TID_val	지지도
T800	11011	1

↓

TID	TID_val	지지도
T100	11010	2
T400	11001	2
T700	10101	1
T900	11100	1

<표 5> 빈발 항목 집합을 구하는 과정

4. 결론

본 논문에서 제안하는 알고리즘은 이진 형식으로 변환할 때 데이터베이스를 스캔하고 나면 더 이상 데이터베이스를 스캔할 필요가 없으므로 데이터베이스를 한 번만 스캔하면 된다. 그리고 데이터베이스를 이진 형식으로 변환하기 때문에 대용량의 데이터베이스를 적은 메모리 공간을 사용하면서 표현할 수 있다. 또한 빈발 항목 집합을 발견하는 과정에서 후보 항목 집합을 생성하지 않는 장점도 있다. 하지만 빈발 항목 집합을 발견하는 과정에서 트랜잭션간의 비교 연산 횟수가 많아지기 때문에 본 논문의 향후 연구과제로는 비교 연산 횟수를 더 줄일 수 있는 방안에 대한 연구가 필요하다.

감사의 글

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었으며(IITA-2008-C1090-0801-0028), 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스 컴퓨팅 및 네트워크 원천 기반기술 개발사업의 08B3-B1-10M 과제로 지원된 것임.

참고문헌

[1] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules", Proceedings of the 20th International Conference on Very Large Databases, pp.487~499, 1994.

[2] H. Mannila, H. Toivonen, A.I. Verkamo, "Efficient Algorithms for Discovering Association Rule", in AAAI Workshop on Knowledge Discovery in Databases, Eds. Usama M. Fayyad and Ramasamy Uthurusamy, pp. 181~192, Seattle, Washington, July 1994.

[3] A. Savasere, E. Omiecinski, S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", In Proceedings of the 21st VLDB Conference, pp.432~444, January 1995.

[4] A. Muller, "Fast sequential and parallel algorithms for association rule mining : A comparison", University of Maryland-College Park CS Technical Report, CS-TR-3515, 76pages, August 1995.

[5] Jong Soo Park, Ming-Syan Chen and Philip S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", Proceedings of ACM SIGMOD, pp.175~186, 1995.

[6] 박종수, 유원경, 홍기형, "연관 규칙 탐사와 그 응용", 정보과학회지, 제16권 제9호, pp.37~44, 1998.

[7] 이재문, 박종수, "복합 해쉬트리를 이용한 효율적인 연관 규칙 탐사 알고리즘", 정보과학회 논문지(B) 제26권 제3호, pp.343~352, 1998.

[8] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining Frequent Patterns without Candidate Generation", In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp.1~12, 2000.

[9] 이형봉, "연관 규칙 탐사 알고리즘의 비교 분석", KSIAM IT series Vol.9 No.1, pp.69~82, 2005.